2020 LGE Code Jam: Online Round 1

Problem A. Card Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.3 sec
Memory limit:	256

Orca and Dolphin love to play a card game together.

Every card is colored red (R), yellow (Y), or blue (B), and it has a digit (0-9) written on it.

There may exist more than one card with the same color and number.

Orca has been losing every game lately, and is beginning to suspect whether Dolphin is cheating by swapping card(s) while they are playing the game.

Orca has a great memory, and therefore can remember the n cards he saw before they played a game as well as the n cards that were played during the game.

However, Orca is not good at checking whether Dolphin cheated or not.

For instance, suppose they are playing a game with $n = 5$ cards as follows:

- R0
- B9
- R5
- Y3
- R2

After the game was played, Orca recalls that the following five cards were played during the game:

- R0
- B8
- R5
- Y3
- R2

Dolphin must have swapped B9 (one of the cards in the original deck) with B8 (which was not present in the original deck)!

You ought to write a program that can help Orca decide whether Dolphin cheated or not.

Input

The first line will contain the number of test cases T.

Each test case will be given by three lines of input.

The first line will contain n, the number of cards in the deck.

The second line will contain n strings, separated by a whitespace, where each string contains two characters. The first character is one of R, Y, and B (color) and the second character is a digit (between 0 and 9). These n strings represent the n cards in the original deck that Orca remembers.

The third line will also contain n strings in the same manner, which represent the n cards that were played during the game.

Output

For each test case, if there is evidence that Dolphin cheated (by swapping a card in the deck with a different card not in the deck), then output "CHEATER".

Otherwise, output "NOT CHEATER".

For each test case, your output must be printed within a single line without quotes.

Constraints

- $1 \leq T \leq 10$
- $1 \leq n \leq 200$

Examples

standard input	standard output
4 5 R0 B9 R5 Y3 R2 R0 B8 R6 Y3 R2 1 R0 R0 3 R1 R0 R0 R0 R1 R1 3 R1 R1 R0 R0 R1 R1	CHEATER NOT CHEATER CHEATER NOT CHEATER

- Test case 1: This was mentioned in the problem statement.
- Test case 2: The list of cards before the game agrees with the list of cards after the game, and thus Dolphin did not cheat.
- Test case 3: Dolphin must have swapped one of the R0 cards with a new R1 card.
- Test case 4: No cards were swapped.

Problem B. Science Fair

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 sec
Memory limit:	256 MB

You are trying to help host the upcoming World Science Fair in your city.

There will be a lot of attendees, and three large gyms will be used for various activities (let us call those gyms A, B, and C).

In order to help organizing the event at each gym, you are trying to assign volunteers to these three gyms – specifically, up to n_A, n_B, and n_C (respectively) volunteers to gym A, gym B, and gym C (respectively).

One volunteer can be assigned to at most one gym, and volunteers do have preferences over which gym(s) they want to be assigned to.

For instance, suppose $n = 4$ and $n_A = n_B = n_C = 1$ (for simplicity, let us label the four volunteers 1, 2, 3, and 4).

- Only volunteer 1 wants to be assigned to gym A.
- Only volunteer 1 wants to be assigned to gym B.
- Both volunteers 2 and 3 want to be assigned to gym C.
- Volunteer 4 does not want to be assigned to any gyms (perhaps this volunteer wants to participate in other volunteering opportunities).

In this case, because $n_C = 1$, we must choose either volunteer 2 or volunteer 3 for gym C.

Volunteer 1 can only be assigned to either gym A or gym B.

Therefore, we can assign at most two volunteers to the gyms in this example.

Given n, n_A, n_B, and n_C and the preferences of n volunteers, write a program that computes how to assign maximum number of volunteers to the gyms.

Input

The first line will contain the number of test cases, T.

For each test case, the first line will contain n and the second line will contain n_A, n_B, and n_C separated by a whitespace.

The following three lines will describe which volunteers want to be assigned to each gym.

The first of the three lines will begin with m_A (the number of volunteers who want to be assigned to gym A), followed by m_A numbers (representing volunteers who are labeled from 1 to n), separated by a whitespace. Likewise, the second line will contain m_B followed by m_B numbers, and the third line m_C followed by m_C numbers.

Output

For each test case, the first line must contain the maximum number of volunteers you can assign to the three gyms, which we call x.

The next x lines must describe how you assign these x volunteers to gyms.

Each line must contain one of the characters (A, B, or C) and the volunteer's number, separated by a whitespace.

If there are multiple assignments with the same number of assigned volunteers, you may find any one of them.

Constraints

- $1 \leq T \leq 10$
- $1 \leq n \leq 10,000$
- $1 \leq n_A, n_B, n_C \leq n$
- $0 \leq m_A, m_B, m_C \leq n$

Subtask 1 (5 points)

- $1 \leq n \leq 100$

Subtask 2 (14 points)

- $1 \leq n \leq 10,000$

Examples

standard input	standard output
3 3 1 1 1 1 1 1 1 2 2 3 4 1 1 1 1 1 1 1 2 2 3 5 1 1 2 1 1 1 5 4 1 2 3 5	2 1 A 2 C 2 2 C 1 B 4 1 A 2 C 5 B 3 C

- Test case 1: It's the same test case as the one described in the problem statement, but without volunteer 4.
- Test case 2: It's the same test case as the one described in the problem statement.
- Test case 3: Although Gym C is very popular, volunteers 1 and 5 should be assigned to a different gym than Gym C in order to maximize the number of assigned volunteers. Similarly to Test Case 2, volunteer 4 is unwilling to be assigned in this case.

Problem C. Team Matching

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.7 sec
Memory limit:	256 MB

You are organizing the upcoming Le Great Hackathon (aka LG Hackathon) featuring cyber security at your company.

Participants will be divided into two teams, Team A and Team B.

Team A will try to hack (attack) a server, and Team B will try to securely defend it.

There are n participants (labeled from 1 to n), and based on previous assessments within the company, every participant's attack/defend skill is quantified.

Each participant Participant i's attack skill is A_i and defend skill is B_i (both are positive integers); these are called A-scores and B-scores, respectively.

Given these n participants, you must divide them into two teams with the following rules:

- Since the two teams must have similar numbers of participants, the difference in the number of participants in the teams should be k or less.
- Every participant must belong to either Team A or Team B.
- Among all possible options, you want to maximize the sum of A-scores of those in Team A and B-scores of those in Team B (that way, the event will be most competitive and fun).

For instance, suppose that $n = 3$ and $k = 1$ with the following A-scores and B-scores:

- $A_1 = 1$ and $B_1 = 100$
- $A_2 = 100$ and $B_2 = 99$
- $A_3 = 80$ and $B_3 = 95$

In this case, because $k = 1$, we must have either 1 person or 2 people in Team A (and 2 or 1 in Team B, respectively).

There are six possibilities for the team assignment:

- Team A [1] Team B [2, 3] with total score $A_1 + B_2 + B_3 = 195$
- Team A [2] Team B [1, 3] with total score $A_2 + B_1 + B_3 = 295$
- Team A [3] Team B [1, 2] with total score $A_3 + B_1 + B_2 = 279$
- Team A [2, 3] Team B [1] with total score $A_2 + A_3 + B_1 = 280$
- Team A [1, 3] Team B [2] with total score $A_1 + A_3 + B_2 = 180$
- Team A [1, 2] Team B [3] with total score $A_1 + A_2 + B_3 = 196$

The second assignment has the largest total score, which is 295.

Given n, k, and everyone's A-score and B-score, find the maximum sum of scores you can obtain while meeting the said rules above.

Input

The first line will contain the number of test cases, T.

For each test case, the first line will contain n and k, separated by a whitespace.

The next line will contain n integers, separated by a whitespace, describing A-scores (A_i) of the participants.

The next line will contain n integers, separated by a whitespace, describing B-scores (B_i) of the participants.

Output

For each test case, you must output the maximum total score you can obtain in a single line.

Constraints

- $1 \leq T \leq 10$
- $3 \leq n \leq 100,000$
- $1 \leq k \leq n - 2$
- $1 \leq A_i, B_i \leq 1,000,000$

Examples

standard input	standard output
4 5 2 1 2 3 4 5 2 3 4 5 6 5 3 1 2 3 4 5 5 4 3 2 1 5 2 1 3 5 7 9 1 2 3 4 5 3 1 1 100 80 100 99 95	18 21 24 295

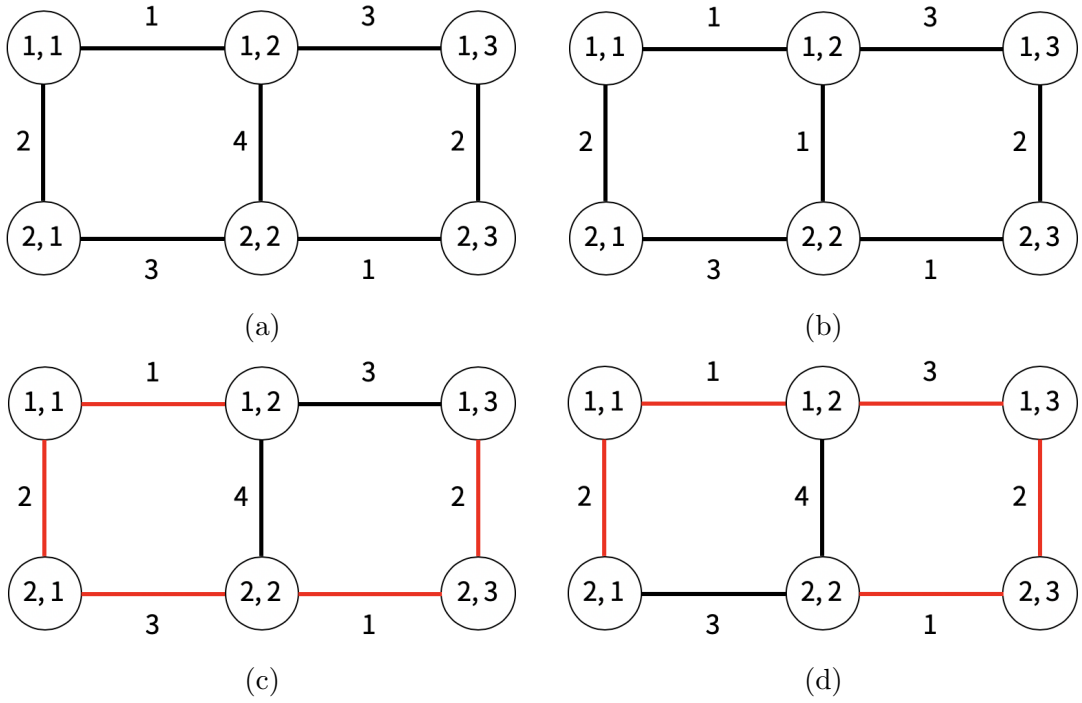
- Test case 1: Team A [1, 2] and Team B [3, 4, 5] will result in the maximum total score of 18.
- Test case 2: Team A [3, 4, 5] and Team B [1, 2] will result in the maximum total score of 21.
- Test case 3: Team A [3, 4, 5] and Team B [1, 2] will result in the maximum total score of 24.
- Test case 4: This test case was mentioned in the problem statement.

Problem D. Grid Network

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 sec
Memory limit:	256 MB

Your consulting firm needs to help a client decide how to setup a grid of supercomputers. These supercomputers are located in a grid (one supercomputer per node in the grid), and your client wants to minimize the cost of installing petabit networks lines so that any pair of supercomputers can communicate with each other (they do not have to be directly connected as they can communicate through other connected supercomputers).

The grid consists of R rows and C columns such that each node contains one supercomputer. A petabit network can be installed between any two adjacent nodes, horizontally or vertically. The cost of connecting a pair of (adjacent) nodes varies, but it is always an integer between 1 and 4 (inclusive). Interestingly, the costs of installing petabit networks for any node would be distinct.



For instance, in Figure (a), there are six nodes, and the costs of installing petabit networks for any specific node are all distinct. In Figure (b), there are six nodes as well, but node (1, 2) and node (2, 2) have non-distinct costs (for instance, node (1, 2) has two petabit networks whose cost is 1).

Given the grid in Figure (a), we can install five petabit networks whose overall cost is 9 (which is optimal), and allow every pair of supercomputers to communicate. There are two ways to do so, as illustrated in Figure (c) and Figure (d). Given a grid network and costs of installing petbit network for adjacent nodes, compute the minimum cost of installing petabit networks so that every pair of supercomputers can directly or indirectly communicate with each other.

Input

The first line will contain the number of test cases, T.

The first line of each test case will contain two numbers, R and C, separated by a whitespace. The following R lines will contain $C-1$ integers each, describing the cost of installing a petabit network between two horizontally adjacent nodes. The following $R-1$ lines will contain C integers each, describing the cost of installing a petabit network between two vertically adjacent nodes.

Output

For each test case, you must output the minimum cost in a single line.

Constraints

- $1 \leq T \leq 10$
- $2 \leq R, C \leq 500$
- $1 \leq$ cost of connecting a pair of nodes ≤ 4

Subtask 1 (4 points)

- $2 \leq R, C \leq 7$
- The total number of edges in a grid is at most 20.

Subtask 2 (9 points)

- $2 \leq R, C \leq 500$

Examples

standard input	standard output
3 2 3 1 3 3 1 2 4 2 2 2 1 1 2 2 3 3 1 2 1 4 4 3 2 3 3 3 2 1	9 4 15

1. Test case 1: This was discussed in the problem statement.
2. Test case 2: There exist two optimal solutions – only one of the two network lines whose cost is 2 should be used.
3. Test case 3: There exist two optimal solutions.

Problem E. Unstable Materials

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.2 sec
Memory limit:	256 MB

Rick the Scientist recently succeeded in synthesizing new ultra-nano scale materials.

He's synthesized n kinds of materials (labeled as 1 to n), and material i's weight is w_i.

Rick wants to present some of these materials at the upcoming Conference on "Unstable Ultra-nano Scale materials".

Yet, since these materials are tiny and unstable, there are some limitations.

Material i may explode if it is put into the same container as another material Q_i.

Given this information, Rick wants to choose some materials that are not going to explode when put into the same container together.

For instance, suppose $n = 3$ and their weights are $w_1 = 3$, $w_2 = 5$, and $w_3 = 1$.

In addition, material 1 reacts to material 2 ($Q_1 = 2$), material 2 to material 1 ($Q_2 = 1$), and material 3 to material 2 ($Q_3 = 2$).

In other words, if materials 1 and 2 are put together, then they would both explode. Material 3 would explode if material 3 and material 2 are put together.

In this case, having material 1 and material 3 is safe, and they can be put together in the same container (whose overall weight is 4).

On the other hand, putting material 2 alone in the container is also safe, and its weight is 5 – therefore this is optimal.

Rick wants to find out the maximum sum of weights of these materials that he can bring to the conference (in a single container) without causing explosions.

Input

The first line will contain the number of test cases, T.

For each test case, the first line will contain the number of materials, n.

Each of the following n lines will contain two numbers, separated by a whitespace, that are w_i and Q_i, respectively.

Output

For each test case, you must output the maximum achievable weight of materials Rick can bring to the conference without causing explosions.

Constraints

- $1 \leq T \leq 10$
- $2 \leq n \leq 100,000$
- $1 \leq w_i \leq 1,000,000$
- $1 \leq Q_i \leq n$
- $Q_i \neq i$

Subtask 1 (6 points)

- $2 \leq n \leq 1,000$

Subtask 2 (21 points)

- $2 \leq n \leq 100,000$

Examples

standard input	standard output
4 3 3 2 5 1 1 2 3 3 2 2 3 2 1 4 1 3 2 3 3 4 4 2 4 2 2 3 1 4 4 3 3	5 3 5 7

- Test case 1: It's optimal to bring material 2 alone.
- Test case 2: It's optimal to bring material 1 alone.
- Test case 3: It's optimal to bring material 1 and material 4.
- Test case 4: It's optimal to bring material 2 and material 3.

Problem F. LG SDK

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 sec
Memory limit:	256 MB

One of your hobbies is to fix bugs in a well-known open source SDK, called the "Le Grate SDK" (aka LG SDK).

Recently, a lot of bugs were discovered, and because of their relationship, you are now solving a fun problem.

There are currently n known bugs (labeled as 1 to n), and you quantified how fun it will be to fix each bug – the fun score.

Specifically, if you fix bug i, then you will have f_i units of fun which is either a positive integer or a negative integer (if it's negative, then you think it'll be boring for you to fix the bug).

In an ideal world, you would only fix the bugs with positive fun scores so as to maximize your overall fun score, but, unfortunately, you realized that fixing some bugs would require other bugs be fixed together.

That is, there are cases where you must fix bug j first in order to fix bug i.

For instance, suppose that $n = 3$ and your fun scores are $f_1 = 5$, $f_2 = -2$, and $f_3 = 3$.

To fix bug 1, you must also fix bug 2, and to fix bug 2, you do not have to fix any other bugs. Lastly, to fix bug 3, you must also fix both bugs 1 and 2.

In this case, if you only fix bug 2, then your overall fun score will be -2, whereas if you fix both bugs 1 and 2 then the overall fun score will be 3.

Lastly, if you fix all three bugs, then the overall fun score will be 6.

Given the number of bugs, their fun scores, and their relationships, find the maximum overall fun score you can achieve.

Input

The first line will contain the number of test cases, T.

For each test case, the first line will contain the number of bugs, n.

The second line will contain n integers, separated by a whitespace, that describe fun scores of the bugs.

The following n lines will contain one or more integers.

The i-th line's first integer will describe the number of other bugs that must be fixed in order to fix bug i.

If x_i is that number, then x_i integers will follow (and be separated by a whitespace).

These x_i bugs will be distinct.

Output

For each test, you must output the maximum achievable overall fun score in a single line.

Constraints

- $1 \leq T \leq 10$
- $2 \leq n \leq 500$
- $1 \leq |f_i| \leq 1,000,000$
- $0 \leq x_i \leq \min(300, n-1)$

Subtask 1 (6 points)

- $1 \leq n \leq 20$

Subtask 2 (21 points)

- $1 \leq n \leq 500$

Examples

standard input	standard output
8 4 2 -3 6 -4 1 2 0 2 2 4 1 2 3 2 -6 3 0 0 2 1 2 3 5 -2 3 1 2 0 2 1 2 3 -2 -3 -4 0 0 0 3 1 -1 2 1 2 1 3 1 1 6 -51 -89 -58 21 -6 35 0 4 1 4 3 5 2 4 6 1 1 4 1 4 6 3 1 1 5 -10 -10 -10 -10 39 0 0 0 0 4 1 2 3 4 7 72 96 -45 -69 -46 65 -70 0 1 1 2 1 2 3 1 2 3 2 1 2 4 1 2 3 5 2 3 5	1 2 6 0 2 5 0 168

- Test case 1: If you fix all bugs, the overall fun score will be 1. You can't just fix bug 1 (which requires fixing bug 2), and fixing bugs 1 and 2 yields fun score of -1. To fix bug 3, you must also fix bugs 2 and 4 (which leads to fun score of -1), and if you also fix bug 1 then the overall score will be 1. This is optimal.
- Test case 2: If you fix all bugs, the overall fun score will be -1. If you only fix bug 1 alone, then the fun score will be 2. To fix bug 3, you must also fix bugs 1 and 2, which results in fun score of -1.
- Test case 3: This was discussed in the problem statement.
- Test case 4: Every bug's fun score is negative, and thus not fixing any bugs would be optimal.
- Test case 5: Bug 1 requires fixing bug 2, which requires fixing bug 3, which in turn requires fixing bug 1. This is a valid case, and you simply need to fix all three bugs or none. If you choose to do the former, then overall score will be 2.
- Test case 6: Fixing bugs 1, 4, and 6 results in the overall score of (-51) + 21 + 35 = 5. Note that both bug 4 and bug 6 require bug 1 to be fixed as well.
- Test case 7: Bug 5 is the only one with positive score, but it requires other bugs be fixed (in which case, the overall score becomes negative). Hence, it's better to fix nothing.
- Test case 8: The optimal solution is to fix bugs 1 and 2, with overall score 168.